## AMENDMENTS TO THE CLAIMS

1. (Currently amended) A computer-implemented method for efficiently parsing input data, comprising:

receiving a data file;

retrieving a stored version of the data file and a ~~template/token~~syntax tree comprising nodes and tokens ~~corresponding to~~representing data within the data file, the tree including at least one static node;

comparing the stored version of the data file with the received data file to identify non-matching content in the received data file;

parsing only the non-matching content of the received data file to form at least one subtree comprising nodes and tokens representing the non-matching content of the received data file;

replacing at least one static node of the ~~template/token~~syntax tree with a token; and

creating a mapping from each token to one of the subtrees.

2. (Canceled)
3. (Canceled)
4. (Previously Presented) The computer-implemented method of claim 1 wherein the data file is a web page.
5. (Previously Presented) The computer-implemented method of claim 1 wherein the data file is an HTML file.
6. (Currently amended) A method for efficiently parsing web pages, comprising:

receiving a first HTML page;

retrieving a cached version of the HTML page and a syntax tree comprising nodes and tokens~~template/ token tree~~ representing data within~~corresponding to~~ the first HTML page, the tree including at least one static node;

comparing the cached version of the HTML page with the received HTML page to identify non-matching content in the received HTML page;

parsing only the non-matching content in the received HTML page to form at least one subtree comprising nodes and tokens representing the non-matching content of the received data file;

replacing at least one static node of the ~~template/tokens~~syntax tree with a token; and

creating a mapping from each token to one of the subtrees.

7. (Canceled)

8. (Currently amended) A method for efficiently parsing HTML pages,

comprising:

receiving a first HTML page;

responsive to a determination that a cached version of the HTML page exists:

retrieving the cached version of the HTML page and a first syntax tree comprising nodes and tokens~~template/token-tree~~ representing data within~~corresponding to~~ the first HTML page, the

first tree including at least one static node;

comparing the cached version of the first HTML page with the received HTML page to identify non-matching content in the received HTML page;

parsing only the non-matching content to form a subtree;

creating a mapping from a token of the first tree to the subtree;

responsive to a determination that the cached version of the HTML page does not exist:

parsing the received HTML page to form a second syntax tree comprising nodes and tokens representing the non-matching content of the received data file ~~template/token-tree~~, the second tree containing at least one static node; and

storing the second tree and the received HTML page.

9. (Currently Amended) A method for providing derivative services comprising:

receiving a first HTML page;

constructing a syntax tree comprising nodes and tokens~~-template/token-tree~~ representing data within~~from~~ the received HTML page, the tree comprising a plurality of nodes;

determining that at least one node of the tree contains static content;

determining that at least one node of the tree contains dynamic content;

replacing the nodes of the tree containing dynamic content with tokens;

parsing the dynamic content to form subtrees representing the dynamic content of the received data file; and

mapping the tokens to the subtrees.

10. (Currently amended) A computer-implemented method of providing

derivative services, comprising:

receiving a request for derivative services content from a customer;

retrieving data from a plurality of primary service providers on behalf of the customer, by:

identifying static content that has been previously retrieved from the primary service providers and stored, and corresponding syntax trees comprising nodes and tokens representing data within the static content~~template/token trees,~~ that have also been stored;

identifying dynamic content that differs from the previously retrieved content;

parsing the dynamic content to form subtrees representing the dynamic content of the received data file;

adding tokens to the ~~template/token~~syntax trees;

mapping the tokens to the subtrees;

creating at least one content page comprising the retrieved data; and

providing the created pages to the customer.

11. (Currently amended) A method for efficiently parsing input data, comprising:

receiving a first data file;

retrieving a stored syntax tree comprising nodes and tokens~~template/token tree~~, the stored ~~template/token~~syntax tree representing data within~~having content associated with~~ the first data file and containing at least one static node and at least one token;

retrieving a second data file, the second data file associated with the first data file;

identifying non-matching content present only in the first data file;

parsing only the non-matching content of the first data file to form at least one subtree comprising nodes and tokens representing the non-matching content of the received data file; and

mapping at least one of the tokens to at least one of the subtrees.

12. (Currently amended) The method of claim 11, further comprising:

responsive to identifying non-matching content present only in the first file:

adding at least one new token to the ~~template/token~~syntax tree.

13. (Currently amended) A system for efficiently parsing input data,

comprising:

at least one virtual browser for retrieving content from primary content servers;

an identification engine, communicatively coupled to the virtual browser for identifying retrieved content;

a cache, communicatively coupled to the virtual browser and the parsing engine, for storing retrieved content and syntax trees comprising nodes and tokens representing data within the retrieved content~~template/token trees~~;

a comparison engine, coupled to the virtual browser for comparing retrieved content with stored content to identify differing content not stored in the cache;

a token master, communicatively coupled to the cache, for allocating new tokens to the virtual browser;

a parsing engine, communicatively coupled to the virtual browser, for parsing content identified by the comparison engine as differing content and forming subtrees comprising nodes and tokens representing the differing content of the received data file ~~from the content~~ and creating a mapping from new tokens to formed subtrees; and

a content server, coupled to the virtual browser.

14. (Canceled)

15. (Currently amended) A computer program product for efficiently parsing input data, the computer program product stored on a computer-readable medium and including instructions for causing a computer to carry out the steps of:

receiving a data file;

retrieving a stored version of the data file and a syntax tree comprising nodes and tokens~~template/token tree~~ representing data within~~corresponding to~~ the data file, the tree including at least one static node;

comparing the stored version of the data file with the received data file to identify non-matching content in the received data file;

parsing only the non-matching content of the received data file to format at least one subtree comprising nodes and tokens representing the non-matching content of the received data file;

replacing at least one static node of the ~~template/token~~syntax tree with a token; and creating a mapping from each token to one of the subtrees.